КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

УДК 004.421.5

АНАЛИЗ И СИНТЕЗ ПРОЦЕДУРЫ ПОРОЖДЕНИЯ КУМУЛЯТИВНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

ANALYSIS AND SYNTHESIS PROCEDURES FOR THE GENERATION OF CUMULATIVE SEQUENCES

В.В. Румбешт
V.V. Rumbesht

Белгородский государственный национальный исследовательский университет, Россия, 308015, Белгород, ул. Победы, 85

Belgorod State National Research University, 85 Pobeda St, Belgorod, 308015, Russia

e-mail: rumbesht@bsu.edu.ru

Аннотация. Статья посвящена решению задач анализа и синтеза процедуры порождения кумулятивной последовательности, используемой в каскадном методе. В этой статье ставится и решается общая задача заполнения таблицы Кэли по как можно меньшему количеству наблюдений за абстрактным устройством, реализующем операцию циклической группы. Полученные результаты применяются для анализа кумулятивной последовательности и реализации порождающей ее процедуры. На основании обобщения подхода к реализации этой процедуры в статье предлагается методика синтеза, позволяющая реализовать в этой процедуре как можно больше допустимых операций.

Resume. The article is devoted to solving problems of analysis and synthesis procedure for generating a cumulative sequence used in a cascade method. In this article, we posed and solved the general problem of filling the table Cayley the fewest number of observations of an abstract device that implements the operation of a cyclic group. The results are used to analyze the cumulative sequence of generating and implementing its procedures. Based on a generalized approach to the implementation of this procedure in the article offers a method of synthesis that allows to realize in this procedure as much as possible admissible operations.

Ключевые слова: каскадный метод, процедура порождения кумулятивных последовательностей, анализ, синтез.
Keywords: cascade method, a procedure of generating a cumulative sequence, analysis, synthesis.

Введение

Одной из основных процедур, применяемых в каскадном методе, является процедура порождения кумулятивной последовательности (процедура порождения КП). Параметрами этой процедуры выступают начальный элемент, принадлежащий носителю циклической группы, и очередной член порождающей последовательности, так же принадлежащий этому множеству. Внутренне состояние процедуры на момент инициализации совпадает с начальным элементом. После каждого обращения к этой процедуре она выдает на выход значение равное результату групповой операции, примененной к внутреннему состоянию и очередному члену порождающей последовательности. При этом очередное внутреннее состояние принимается равным выходу [1]. В каскадном методе процедура порождения КП применятся на каждом уровне преобразований и формирует выходные последовательности этих уровней.

В работе [2] введено понятие конфигурации каскадов, которое предполагает, что на каждом уровне преобразования в процедуре порождения КП может применяться любая операция из $\Theta_{\neq}$ - множества так называемых попарно неконгруэнтных операций, которое содержит $\frac{(N-1)!}{\varphi(N)}$ элементов, где N - порядок группы, φ - функция Эйлера. Уже при относительно небольших значениях N

количество элементов Θ_{τ} оказывается велико даже для их полного перечисления, не говоря уже о реализации всех этих операций.

Для дальнейшего развития и реализации каскадного метода актуальными являются следующие задачи:

- задача анализа процедур порождения КП, которая состоит в получении описания операций любой допустимой, но заранее неизвестной конфигурации каскадов по последовательности, порождаемой каскадным методом;
- задача синтеза таких процедур порождения КП, которые поддерживают как можно больше операций из Θ_{τ}.

Целью данной статьи является поиск решения задачи анализа процедуры порождения КП по ее непосредственному выходу, обобщение полученных результатов и создание методики синтеза процедур порождения КП, позволяющей реализовать в них как можно большее количество неконгруэнтных операций.

Заполнение таблицы Кэли по наблюдениям за процедурой, реализующей операцию циклической группы

Прежде чем приступить к анализу процедуры порождения КП, сформулируем и решим более общую задачу. Пусть имеется абстрактное устройство (процедура), имеющее два входа и один выход, которое рассматривается как черный ящик. Об этом устройстве известно только то, что оно реализует операцию циклической группы $<U,\otimes>$ порядка N. Имеется возможность подавать элементы U на его входы и получать с выхода результат его работы. Требуется, используя как можно меньше обращений к устройству, заполнить таблицу Кэли групповой операции.

Самый простой способ заполнения таблицы Кэли заключается в подаче на входы устройства всех возможных пар элементов U с последующим снятием результата и помещения его в соответствующую ячейку таблицы. Очевидно, что количество обращений к устройству в этом случае составит N^2.

Учитывая, что циклическая группа является коммутативной, и то, что каждая строка таблицы Кэли групповой операции представляют собой перестановку элементов U, то есть имеет $N-1$ степеней свободы, количество обращений к устройству можно сократить до $\sum_{n=1}^{N-1} n = \frac{N^2-N}{2}$. Но и это решение не удовлетворяет условию поставленной задачи.

Приемлемое решение можно получить, если заполнить строку таблицы Кэли, соответствующую одному из образующих элементов группы $<U,\otimes>$. Для заполнения такой строки потребуется $N-1$ обращение к устройству.

Идея этого решения обусловлена тем, что по теореме Кэли [3] любая конечная группа $<U,\otimes>$ изоморфна некоторой подгруппе симметрической группы $S(U)$. Такой изоморфизм определяется следующим образом: каждому элементу $u \in U$ сопоставляются подстановка π_u, такая что $\forall x \in U : \pi_u(x) = u \otimes x$. Если $g \in U$ является образующим элементом циклической группы $<U,\otimes>$, то и π_g - образующий элемент циклической подгруппы симметрической группы $S(U)$.

Подстановка π_g полностью определят соответствующую строку таблицы Кэли операции $\otimes$. Возводя π_g в различные степени от 2 до N (применяется операция умножения подстановок), можно получить все остальные элементы указанной подгруппы симметрической группы $S(U)$ и, следовательно, все недостающие строки таблицы Кэли для операции $\otimes$.

По условию задачи нам неизвестно, какие элементы группы $<U,\otimes>$ являются образующими, и заполнение таблицы Кэли придется выполнять в два этапа: сначала путем обращения к нашему устройству найти образующий элемент $g \in U$ и построить подстановку π_g, и только затем реализовать указанную идею. На рис. 1a и рис. 1b приведены алгоритмы выполнения этих этапов.

Алгоритм *CreateSubst* (рис. 1a) выполняет построение подстановки $\pi : U \to U$, такой что $\forall u \in U : \pi(u) = g \otimes u$, где g - образующий элемент группы $<U,\otimes>$. В качестве входа он принимает множество U и операцию $\otimes$ (если угодно – наше абстрактное устройство). Результатом его работы являются g и π.

На первом шаге алгоритма *CreateSubst* выполняется пометка всех элементов U как необработанных. Далее выполняется итерационный процесс произвольного выбора необработанного элемента $g \in U$ и попытка полного определения подстановки π с помощью операции $\otimes$. При этом

каждый элемент, являющийся результатом операции $\otimes$, помечается как обработанный. Процесс определения подстановки π (возможно не полного) завершается тогда, когда результатом операции окажется выбранный элемент g. В ходе этих действий все элементы, порядок которых делит порядок элемента g, окажутся помеченными как обработанные. После чего итерационный процесс выбора необработанного элемента и попытки построения для него подстановки продолжается. Если выбранный элемент g окажется образующим группы $<U,\otimes>$, то подстановка π будет определена полностью, и все элементы U будут помечены как обработанные, что обеспечивает выход из цикла и завершение алгоритма.

Алгоритм *CreateCaylay* (рис. 1b) выполняет заполнение таблицы Кэли для операции $\otimes$. Входными данными этого алгоритма являются множество U и его мощность N, образующий элемент $g \in U$ и соответствующая ему подстановка π. Результатом его работы является T - таблица Кэли.

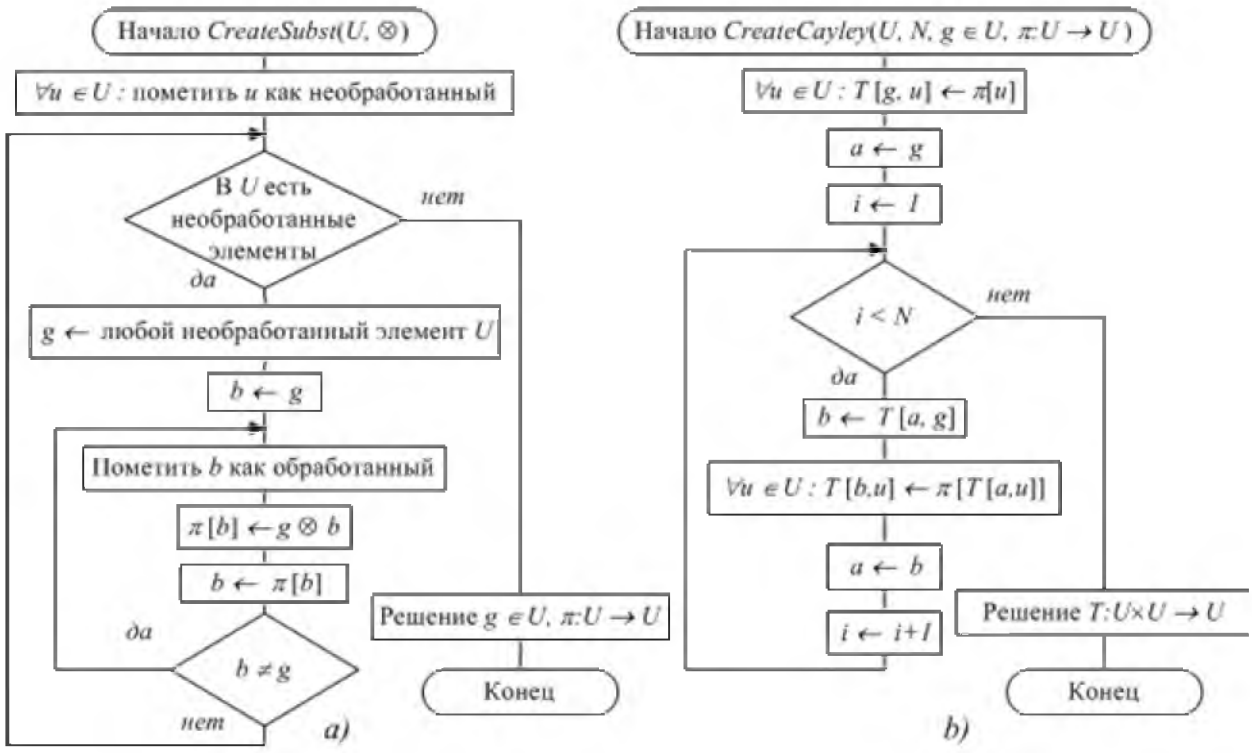

Рис. 1. Этапы заполнения таблицы Кэли:
a) алгоритм построения подстановки $\pi : U \to U$, такой, что $\forall u \in U : \pi(u) = g \otimes u$,
где g - образующий элемент группы $<U,\otimes>$;
b) алгоритм заполнения таблицы Кэли по g и π

Fig. 1. The steps of filling the table Cayley:
a) the algorithm for constructing the substitution $\pi : U \to U$,
such that $\forall u \in U : \pi(u) = g \otimes u$, where g - generator of the group $<U,\otimes>$;
b) the algorithm of filling out the Cayley table for g and π

Первый шаг алгоритма *CreateCaylay* состоит в заполнении строки T, соответствующей элементу g. Для этого применяется подстановка π. Далее выполняется $N-1$ итерация для всех оставшихся строк T. На каждой из этих итераций применяются две вспомогательных переменные a и b, содержащие элементы U, соответствующие предыдущей и текущей заполняемым строкам. Изначально полагается, что a содержит g. Каждая итерация начинается с помещения в b содержимого $T[a,g]$, что дает эффект вычисления очередной степени g. Затем заполняется строка T, соответствующая элементу b. Для заполнения этой строки применяется композиция подстановки π и подстановки, соответствующей элементу a, что аналогично вычислению очередной степени π. Итерация завершается помещением в a содержимого b. После выполнения всех итераций таблица Кэли оказывается заполненной.

Из приведенного выше описания видно, что обращения к устройству, реализующему групповую операцию, сконцентрированы в алгоритме *CreateSubst*. Количество этих обращений зависит от того, какие элементы будут выбираться для попыток построения подстановки. Можно выделить лучший и худший случай для выполнения этого алгоритма. В лучшем случае в качестве g сразу будет выбран образующий элемент группы $<U,\otimes>$ и понадобится N обращений. В худшем случае первым будет выбран нейтральный элемент (элемент первого порядка) и затем элементы по возрастанию их порядков, вплоть до образующего элемента. Очевидно, что в худшем случае потребуется количество обращений к устройству равное $\sum_{d|N} d$, где d - делитель порядка группы и суммирование выполняется по всем таким делителям.

Анализ кумулятивной последовательности и реализация порождающей ее процедуры

Как и в предыдущем случае будем считать, что имеется абстрактное устройство без входов и с одним выходом, рассматриваемое как черный ящик. Далее будем называть его генератором. О нем известно, что он реализует каскадный метод порождения периодических последовательностей. Известно множество U, над которым строятся последовательности, его мощность N и количество уровней преобразований в методе k. Имеется возможность получать члены последовательности с выхода генератора строго в порядке их порождения. Требуется установить групповую операцию $\otimes$, применяемую в процедуре порождения КП последнего уровня преобразования, и оценить, сколько для этого потребуется обращений к генератору.

Пусть $X_{\rightarrow} = (x_1, x_2, \ldots x_i, \ldots)$ - последовательность, формируемая на выходе генератора. Согласно [1] эта последовательность обладает следующими структурными свойствами: ее период составляет N^k, характеристическая функция ее периодического отрезка имеет вид $\forall u \in U : \chi_{[X_{\rightarrow}]}(u) = N^{k-1}$, то есть на периодическом отрезке $[X_{\rightarrow}]$ присутствуют все элементы U, и каждый из них встречается N^{k-1} раз. Более того, в [1] показано, что члены $X_{\rightarrow}$ обладают следующим свойством: для всех натуральных i: $x_{i+N^{k-1}} = x_i \otimes g$, где g – параметр k-го уровня преобразования, являющийся образующим в группе $<U,\otimes>$.

Указанные свойства $X_{\rightarrow}$ могут быть использованы для определения подстановки π, такой, что $\forall u \in U : \pi(u) = g \otimes u$. На рис. 2 приведен алгоритм построения этой подстановки.

Алгоритм *CreateSubst1* (рис. 2) выполняет построение подстановки $\pi : U \to U$, такой что $\forall u \in U : \pi(u) = g \otimes u$, где g – образующий элемент группы $<U,\otimes>$. В качестве входа он принимает множество U, его мощность N, период последовательности, порождаемую генератором $\tau = N^k$, и собственно сам генератор $G()$ Результатом его работы является π. На первом шаге алгоритма *CreateSubst1* выполняется пометка всех элементов U как необработанных, вычисление величины $n = \frac{\tau}{N} = N^{k-1}$ и заполнение вспомогательного массива x первыми n членами последовательности. Для заполнения массива производится обращение к генератору $G()$. Далее выполняется итерационный процесс определения подстановки π до тех пор, пока все элементы U окажутся помеченными как обработанные. В ходе этого процесса выполнятся цикл по перебору элементов массива x в направлении от первого к последнему, который завершается, если будут перебраны все его элементы, либо досрочно – когда все элементы U окажутся помеченными как обработанные. Тело цикла содержит обращение к генератору $G()$ и помещение результата его работы во вспомогательную переменную a. Затем выбирается очередной элемент массива $x[i]$. Если он является необработанным элементом U, то помечается как обработанный и полагается что $\pi[x[i]]$ равно a. Далее, независимо от предыдущего условия, в элемент $x[i]$ помещается значение a и выполняется переход к следующему элементу массива. По окончании выполнения цикла итерационный процесс определения подстановки продолжается. Когда все элементы U будут обработаны, итерационный процесс завершится и подстановка π будет полностью определена.

В алгоритме *CreateSubst1* количество обращений к генератору зависит от последовательности, им порождаемой. Можно выделить лучший и худший случаи выполнения алгоритма. В лучшем случае первые N члены последовательности пробегают все множество U. Очевидно, что для полного определения подстановки потребуется $N^{k-1} + N$ обращений к генератору. В худшем случае пер-

вые N^{k-1} члены последовательности равны между собой. Количество обращений к генератору в данном случае составит $N^k + 1$.

Следует отметить, что по условию задачи нам не известен g-параметр k-го уровня преобразования каскадного метода. Поэтому определить групповую операцию для этого уровня можно лишь с точностью до класса эквивалентности конгруэнтных операций [2]. Во введении к данной статье отмечено, что на каждом уровне преобразования в процедуре порождения КП может применяться любая операция из множества попарно неконгруэнтных операций. Это множество формируется путем включения в него по одному элементу из каждого класса эквивалентности конгруэнтных операций [2]. Какая конкретно операция будет выбрана из класса эквивалентности – не принципиально. Следовательно, операцию можно выбрать произвольно, а для заполнения таблицы Кэли можно воспользоваться алгоритмом *CreateCaylay* (рис. 1b) с параметрами U, N, g и π, где π - подстановка сформированная алгоритмом *CreateSubst1*, а g-любой элемент U.

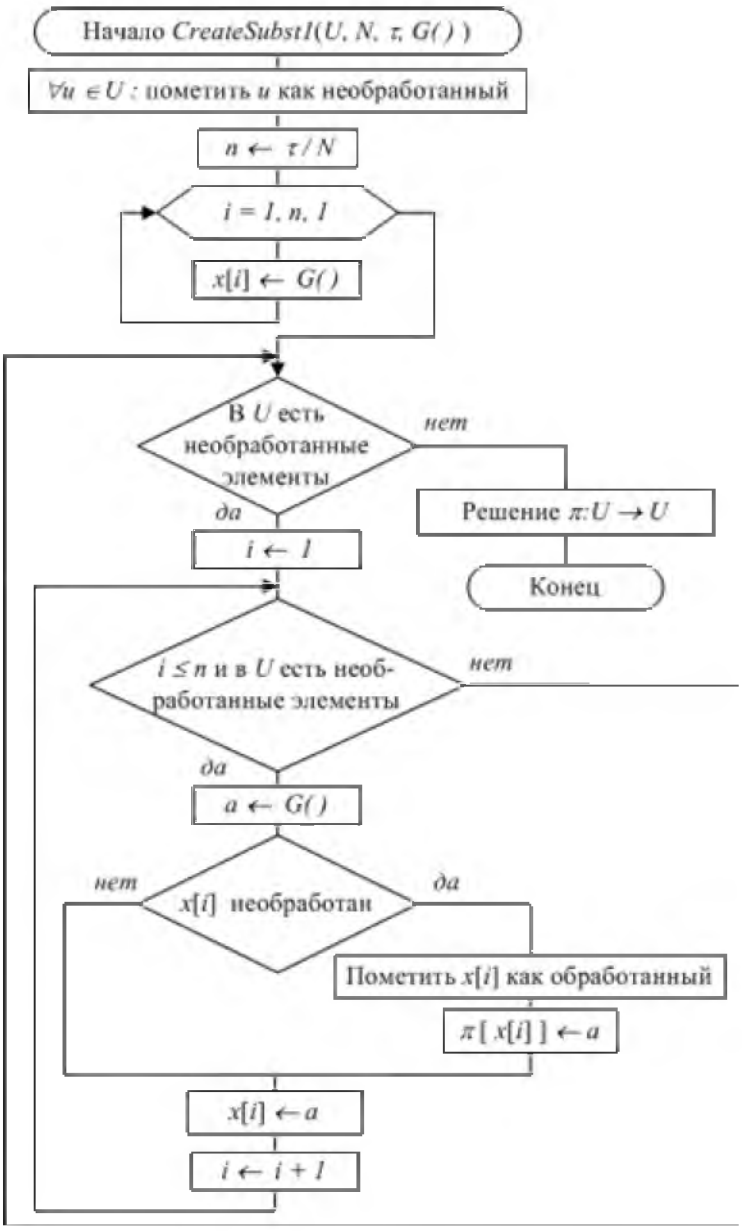

Рис. 2. Алгоритм построения подстановки $\pi : U \to U$ по кумулятивной последовательности
Fig. 2. The algorithm for constructing the substitution $\pi : U \to U$ on the cumulative sequence

Реализация процедуры порождения КП в основном сводится к реализации групповой операции. Вариант реализации с прямым использованием таблицы Кэли является не самым лучшим решением. Хотя он и хорош с точки зрения временной сложности (имеет временную сложность по-

рядка $O(1)$), но для хранения таблицы требуется N^2 ячеек памяти. Поэтому требуется найти более эффективное решение.

Все циклические группы заданного порядка N изоморфны между собой. В частности все они изоморфны аддитивной группе кольца вычетов по модулю N. Обозначим эту группу как $<Z_N,+>$, где $Z_N=\{0,1,\ldots,N-1\}$, "+" – операция сложения по модулю N. Рассмотрим $\psi: U \to Z_N$ - изоморфизм группы $<U,\otimes>$ на $<Z_N,+>$. Прообраз элемента $u \in U$ на множестве Z_N будем называть индексом.

Поскольку биекция ψ является изоморфизмом групп $<U,\otimes>$ и $<Z_N,+>$, имеем $\forall a,b \in U:$ $\psi(a \otimes b)=\psi(a)+\psi(b)$. Отсюда операцию группы $<U,\otimes>$ можно определить как $\forall a,b \in U:\ a \otimes b=\psi^{-1}(\psi(a)+\psi(b))$, где ψ^{-1} - биекция обратная к ψ.

Отсюда непосредственно следует вариант реализации операции $\otimes$ с помощью индексов. Введем так называемую таблицу индексов. Для этого присвоим номера от 0 до $N-1$ всем элементам множества U с помощью функции-нумератора $Num: U \to Z_N$. Тот факт, что для элемента $u \in U$: $Num(u)=i$, будем обозначать u_i. Таблицу индексов будем представлять в виде массива Ind из N чисел. Элементы массива нумеруются от 0 до $N-1$. В таблице индексов элемент массива $Ind[i]$ содержит $\psi(u_i)$. Вычисление результата операции $\otimes$ сводится к двум обращениям к таблице индексов по номерам элементов-операндов операции, вычислении суммы по модулю N их индексов, с последующим линейным поиском в массиве Ind номера элемента соответствующего полученной сумме. Поскольку Ind содержит все числа 0 до $N-1$, такой поиск всегда будет успешен, и его результат – есть номер результата операции $\otimes$. Очевидно, что временная сложность этого варианта реализации групповой операции имеет порядок $O(N)$ и, при этом, требуется всего N ячеек памяти для хранения таблицы индексов.

Заполнение таблицы индексов может быть выполнено с помощью таблицы Кэли групповой операции. По таблице Кэли определяется нейтральный элемент группы и ему назначается индекс 0. Из множества образующих элементов группы $<U,\otimes>$ произвольным образом выбирается один элемент g_{Base}, так называемый базовый образующий, и ему назначается индекс 1. Далее с помощью таблицы Кэли g_{Base} возводится в степени от 2 до $N-1$, и соответствующим степеням g_{Base} назначается индекс равный показателю степени.

Методика синтеза процедуры порождения кумулятивных последовательностей

Предложенный выше способ реализации групповой операции с помощью таблицы индексов предполагает, что должна быть задана таблица Кэли. Кроме этого получается, что в данной реализации имеем следующую ситуацию: одна таблица индексов – одна операция. Это не совсем удобно, если требуется единообразно и с минимальными затраченными ресурсами реализовать большое количество операций, которые могут быть применены для конфигурирования каскадного метода. В этом разделе будет предложена методика, позволяющая с использованием одной таблицы индексов реализовать не одну, а множество операций.

Для начала определим, какими свойствами должны обладать таблицы индексов, задающие неконгруэнтные операции.

Несложно заметить, что любая таблица индексов есть подстановка на Z_N. Всего таких подстановок N, а различных операций, образующих из множества U циклическую группу порядка N, существует $\frac{N}{\varphi(N)}$ [2]. Дело в том, что одной и той же операции соответствуют $\varphi(N)$ таблиц индексов. Пусть $Ind_{\otimes}$ и $Ind_{\overline{\otimes}}$ – таблицы индексов, соответствующие операциям $\otimes$ и $\overline{\otimes}$. Очевидно *утверждение 1*: операции $\otimes$ и $\overline{\otimes}$ не равны, если $\exists u_i,u_j \in U:\ Ind_{\otimes}[i]=Ind_{\overline{\otimes}}[i]=1$ и $Ind_{\otimes}[j] \neq Ind_{\overline{\otimes}}[j]$.

Кроме этого истинно *утверждение 2*: достаточным условием не конгруэнтности или равенства операций $\otimes$ и $\overline{\otimes}$ является $\exists u_k \in U:\ Ind_{\otimes}[k]=Ind_{\overline{\otimes}}[k]=0$. Действительно, когда операции $\otimes$ и $\overline{\otimes}$ являются конгруэнтными, но не равными, в группе $<U,\otimes>$ существуют отличные от нейтрального элемента p и p^{-1}, где p - параметр конгруэнтности $\otimes$ и $\overline{\otimes}$. Если p является параметром конгруэнтности $\otimes$ и $\overline{\otimes}$, то p^{-1} должен быть нейтральным элементом в $<U,\overline{\otimes}>$ [2]. Это противоречит

равенству нейтральных элементов групп $<U,\otimes>$ и $<U,\overline{\otimes}>$, следовательно, и условию $\exists u_k \in U$: $Ind_{\otimes}[k] = Ind_{\overline{\otimes}}[k] = 0$.

Из утверждений 1 и 2 следует, что операции $\otimes$ и $\overline{\otimes}$ неконгруэнтны, если

$$\exists u_i, u_j, u_k \in U: \; Ind_{\otimes}[i] = Ind_{\overline{\otimes}}[i] = 1 \text{ и } Ind_{\otimes}[j] \neq Ind_{\overline{\otimes}}[j] \text{ и } Ind_{\otimes}[k] = Ind_{\overline{\otimes}}[k] = 0. \quad (1)$$

Таким образом, приняв, например, $u_i = u_1$ и $u_k = u_0$ можно получить $(N-2)!$ различных таблиц индексов, каждая из которых соответствует своей уникальной операции, и все эти операции являются попарно не конгруэнтными. Следует отметить, что этими таблицами индексов можно задать только $\frac{\varphi(N)}{N-1}$ часть множества $\Theta_{\neq}$, поскольку условие (1) является лишь достаточным для неконгруэнтности операций.

Обозначим символом $S(Z_N \setminus \{0,1\})$ подгруппу симметрической группы $S(Z_N)$, элементами которой являются подстановки, переводящие 0 в 0 и 1 в 1, то есть имеющие как минимумдве неподвижные точки. Как было показано, элементы $S(Z_N \setminus \{0,1\})$ – суть таблицы индексов, задающие попарно неконгруэнтные операции.

Идея единообразной реализации множества попарно неконгруэнтных операций с помощью одной таблицы индексов заключается в выделении какой-либо циклической подгруппы $S(Z_N \setminus \{0,1\})$, которую обозначим как $S(Z_N \setminus \{0,1\})$. В этой подгруппе фиксируется некоторый образующий элемент – базовая таблица индексов, с помощью которой находятся все остальные таблицы индексов: элементы $S(Z_N \setminus \{0,1\})$. При таком подходе в памяти можно хранить только базовую таблицу, а значения остальных подстановок могут быть вычислены так, как показано в алгоритме исполнения групповой операции на рис. 3. Таким образом, с помощью одной базовой таблицы индексов можно реализовать n попарно неконгруэнтных операций, где n - порядок $S(Z_N \setminus \{0,1\})$.

Алгоритм *ExeOperation* (рис. 3) находит значение номера элемента $u_c = u_a \otimes u_b$, где $\otimes$ - операция группы $<U,\otimes>$, таблица индексов которой принадлежит $S(Z_N \setminus \{0,1\})$. Входными данными этого алгоритма являются: a и b - номера элементов $u_a \in U$ и $u_b \in U$; N - порядок $<U,\otimes>$; Ind - базовая таблица индексов (образующий элемент $S(Z_N \setminus \{0,1\})$); n - порядок группы $S(Z_N \setminus \{0,1\})$; m - номер операции (целое число от 0 до $n-1$, представляющее собой показатель степени подстановки, в которую возводится Ind для получения таблицы индексов выполняемой операции). Результатом выполнения алгоритма является c - номер элемента $u_c \in U$.

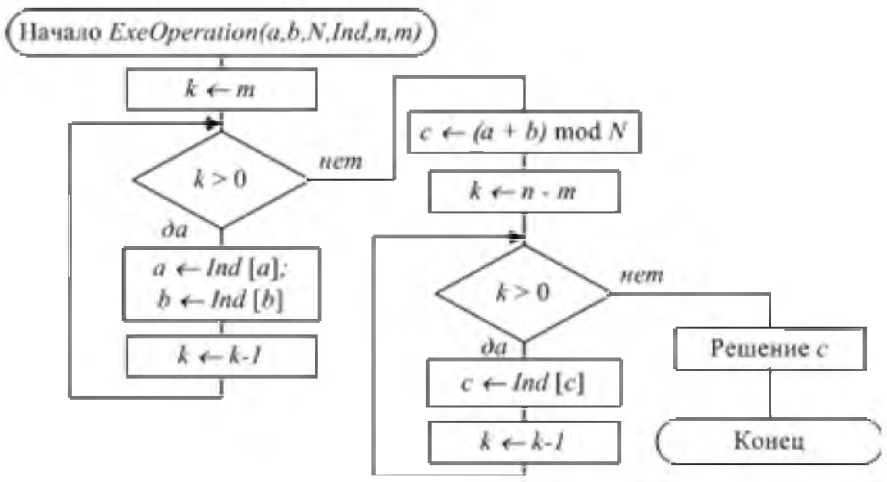

Рис. 3. Алгоритм исполнения групповой операции
Fig. 3. The algorithm execution group operation

На первом этапе выполнения алгоритма *ExeOperation* значения a и b заменяются их образами в подстановке Ind^m, что реализовано в виде цикла из m итераций, каждая из которых состоит в замене значений a и b на значения $Ind[a]$ и $Ind[b]$ соответственно. Затем к полученным значениям a и b применяется операция "сумма по модулю N", результат которой помещается в переменную c. На завершающем этапе значение c заменяется ее прообразом в подстановке Ind^m, или,

что тоже самое – ее образом в подстановке обратной к Ind^m. Это действие реализовано циклом по замене c на $Ind[c]$, который выполняется $n-m$ раз.

Выполнение алгоритма *ExeOperation* требует N ячеек памяти для хранения базовой таблицы индексов, а его временная сложность имеет порядок $O(n)$.

Основная сложность в переложенном подходе состоит в нахождении базовой таблицы индексов, позволяющей реализовать как можно большее количество операций.

Для решения этой задачи рассмотрим следующие соображения, основанные на материале из источника [3]. Отметим, что любая подстановка $\pi \in S(Z_N \setminus \{0,1\})$ является образующим элементом некоторой циклической подгруппы $S(Z_N \setminus \{0,1\})$. Порядок этой подгруппы совпадает с порядком π. Кроме этого, любая подстановка представляется произведением независимых циклов. Все множество подстановок $S(Z_N \setminus \{0,1\})$ может быть разбито на классы эквивалентности, согласно структуре циклов, образующих произведение его элементов. Подмножество из $S(Z_N \setminus \{0,1\})$ образует класс $K = (k_1 + 2, k_2, \ldots, k_{N-2})$, если для каждой подстановки из этого подмножества количество независимых циклов длины $l \in \{2, 3, \ldots, N-2\}$ составляет k_l. В силу специфики элементов $S(Z_N \setminus \{0,1\})$, у всех классов эквивалентности количество циклов длины 1 составляет $k_1 + 2$. Очевидно, что для любого класса эквивалентности подстановок $(k_1 + 2, k_2, \ldots, k_{N-2})$ выполняется равенство

$$\sum_{l=1}^{N-2} k_l \cdot l = N - 2. \qquad (2)$$

При этом каждое решение уравнения (2) соответствует одному классу, а все решения покрывают все возможные классы эквивалентности подстановок из $S(Z_N \setminus \{0,1\})$.

Порядок подстановки равен наименьшему общему кратному длин циклов, определяющих класс эквивалентности, которому она принадлежит. То есть решение уравнения (2) определяет цикловую структуру и позволяет найти порядок элементов соответствующего класса эквивалентности подстановок. Поскольку нас интересует реализация как можно большего количества операций, в качестве базовой таблицы индексов следует выбрать подстановку, имеющую максимальный порядок. Для этого требуется вычислить функционал

$$n = \max_{\sum_{l=1}^{N-2} k_l \cdot l = N-2} \left(lcm\left(j^{f(k_j)} \right) \right), \qquad (3)$$

где $f(k) = \begin{cases} 0, if\ k < 1; \\ 1, if\ k \geq 1. \end{cases}$, а максимум находится по всем решениям уравнения (2). В работе [3] описан алгоритм нахождения этих решений методом последовательного лексикографического перечисления. Для нашего случая потребуется не сложная модификация данного алгоритма. Модификация состоит в подсчете наименьшего общего кратного для очередного решения уравнения (2), сохранении текущего максимального значения *lcm* и соответствующего решения. В результате будет получена цикловая структура подстановок из $S(Z_N \setminus \{0,1\})$, имеющих максимальный порядок. Остается только подобрать или построить подстановку, удовлетворяющую этой цикловой структуре. На рис. 4 приведен алгоритм построения базовой таблицы индексов по заданной цикловой структуре и перестановке элементов множества Z_N.

Входными параметрами алгоритма *CreateBaseInd* (рис. 4) являются: N – количество элементов в таблице индексов; P – массив размерности N, в котором $P[0] = 0$ и $P[1] = 1$, а остальные элементы – некоторая перестановка оставшихся чисел из Z_N; K – массив размерности $N-2$, элементы которого задают цикловую структуру класса эквивалентности $(k_1 + 2, k_2, \ldots, k_{N-2})$ так, что $K[0] = k_1 + 2$, $K[1] = k_2$ и т.д., $K[N-3] = k_{N-2}$. Результат выполнения данного алгоритма есть $Ind \in S(Z_N \setminus \{0,1\})$ – базовая таблица индексов. В этом алгоритме вычисляется произведение независимых циклов для подстановки Ind. При этом массив P рассматривается как запись произведения независимых циклов. Вспомогательные переменные b и e применяются для указания начала и конца очередного цикла из этого произведения, а их значения определяются по описанию цикловой структуры из массива K. Перестановка в массиве P может быть порождена случайным образом с помощью алгоритма Фишера–Йетса [4].

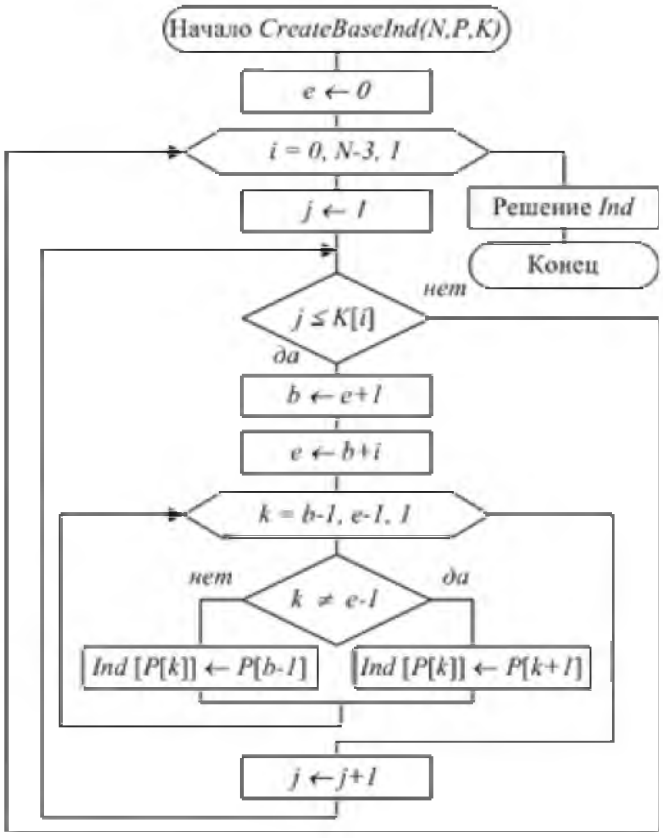

Рис. 4. Алгоритм построения базовой таблицы индексов
Fig. 4. The algorithm for constructing the base table indexes

Сформулируем окончательно методику синтеза процедуры порождения КП. Исходными данными методики является N – порядок группы, над которой порождается последовательность. Процедура порождения КП реализует алгоритм исполнения групповой операции (рис. 3). Параметр $m \in \{0, 1, \ldots, n-1\}$ этого алгоритма используется для конфигурирования и содержит номер операции. Параметр n – количество операций определяется при вычислении функционала (3) путем нахождения решений уравнения (2) по модифицированному алгоритму последовательного лексикографического перечисления. Одновременно с этим устанавливается цикловая структура базовой таблицы индексов (параметра *Ind*). Затем с помощью алгоритма Фишера–Йетса порождается случайная перестановка элементов множества $\{2, 3, \ldots, N-1\}$, которая вместе с полученной цикловой структурой используется в алгоритме построения базовой таблицы индексов *Ind* (рис. 4).

Заключение

Как было отмечено ранее, оценка временной сложности алгоритма исполнения групповой операции *ExeOperation* составляет $O(n)$, где n – количество операций, реализуемых алгоритмом. Опыт применения методики синтеза процедуры порождения КП показал, что функционал (3) растет сложным нелинейным образом в зависимости от N, причем довольно быстро (но не быстрее, чем 2^{N-2}). Так, например, при $N=8$ значение n равно 6, при $N=16$ значение n равно 84, при $N=32$ значение n равно 4620, при $N=64$ значение n уже равно 1141140, при $N=128$ значение n превосходит $6{,}75 \cdot 10^9$, а при $N=256$ значение n уже превосходит $3{,}82 \cdot 10^{15}$. Факт быстрого роста количества операций n свидетельствует об ограничении в применении методики синтеза процедуры порождения КП при стремлении реализовать максимально возможное количество операций.

Список литературы
References

1. Румбешт В.В. 2014. Каскадный метод порождения периодических последовательностей над элементами циклической группы. Научные ведомости БелГУ. Серия: История. Политология. Экономика. Информатика. № 8 (179) Выпуск 30/1: 103–112.

Rumbesht V.V. 2014. Kaskadnyj metod porozhdenija periodicheskih posledovatel'nostej nad jelementami ciklicheskoj gruppy. Nauchnye vedomosti BelGU. Serija: Istorija. Politologija. Jekonomika. Informatika. № 8 (179) Vypusk 30/1: 103–112.

2. Румбешт В.В., Ядута А.З. 2015. Анализ применения конкретных групп в каскадном методе. Научные ведомости БелГУ. Серия: Экономика. Информатика. № 7 (204) Выпуск 34/1: 105–115.

Rumbesht V.V., Jaduta A.Z. 2015. Analiz primenenija konkretnyh grupp v kaskadnom metode. Nauchnye vedomosti BelGU. Serija: Jekonomika. Informatika. № 7 (204) Vypusk 34/1: 105–115.

3. Кофман А. 1975. Введение в прикладную комбинаторику. М., Наука, 480.

Kofman A. 1975. Vvedenie v prikladnuju kombinatoriku. M., Nauka, 480.

4. Кнут Д. 2007. Искусство программирования, том 2. Получисленные алгоритмы. М., Вильямс, 832.

Knut D. 2007. Iskusstvo programmirovanija, tom 2. Poluchislennye algoritmy. M., Vil'jams, 832.